This listing of claims will replace all prior versions, and listings, of claims in the application.

Listing of Claims:

1. (Currently Amended) A method of automating ~~the~~ generation of a vendor-provided manifest that governs ~~the~~ execution of a software object distributed by the vendor, the method comprising:

creating a manifest configuration file (MCF) that provides a description of requirements to be embodied in the vendor-provided manifest, the description including an identity of a key file that contains a cryptographic key, wherein providing said identity of the key file eliminates the need to manually insert cryptographic key data into the vendor-provided manifest;

parsing the MCF to create a generic representation containing substance specified in the MCF; and

providing the generic representation to a manifest generation tool that reads the generic representation, retrieves said cryptographic key to obtain cryptographic key data for insertion into the vendor-provided manifest, and generates the vendor-provided manifest in an eXtensible Rights Markup Language (XRML) format based on the requirements, said manifest comprising one or more rules imposed by the vendor to enable a security component to impose a permeable barrier for ensuring integrity of an address space that is used in a computer for executing the software object, the one or more rules incorporating a list of acceptable and unacceptable modules, wherein the acceptable modules are permitted to pass through the permeable barrier and are executed in the address space of the computer and the unacceptable modules are prevented from passing through the permeable barrier and unconditionally barred from being executed in the address space of the computer.

2. (Previously Presented) The method of claim 1, wherein said MCF identifies the acceptable and unacceptable modules, and wherein generating the manifest comprises

including, in said manifest, the identities of the acceptable and unacceptable modules identified in said MCF.

3-5. (Canceled)

6. (Previously Presented) The method of claim 2, wherein said MCF indicates whether said manifest will contain hashes for identifying the unacceptable modules.

7. (Previously Presented) The method of claim 1, wherein at least one of said acceptable modules comprises a key, and wherein said MCF indicates that the at least one of said acceptable modules signed with said key may be loaded into said address space, and wherein generating said manifest comprises:

retrieving said key by using a filename provided in said MCF; and

including said key in said manifest.

8. (Previously Presented) The method of claim 1, wherein generating said manifest comprises:

computing a hash of at least one of said unacceptable modules; and

including said hash in said manifest.

9. (Canceled)

10. (Previously Presented) The method of claim 1, further comprising:

receiving a key associated with at least one of a) a vendor or b) a distributor of said software object;

signing said manifest with said key to produce a digital signature; and

including said digital signature in said manifest.

11. (Previously Presented) The method of claim 1, further comprising:

using a hardware security module to sign said manifest, said hardware security module being adapted to apply a key associated with at least one of a) a vendor or b) a distributor of said software object without revealing said key outside said hardware security module.

12. (Previously Presented) A computer-readable storage medium encoded with computer-executable instructions to perform a method of generating a manifest that governs the execution of a software object distributed by a vendor, the method comprising:

generating a file containing a high-level description of the manifest using human-readable syntax, wherein the high-level description comprises a vendor-specified policy configured to preclude loading of a rogue module into an address space of a computer in which the software object is to be executed;

parsing the file by eliminating at least a portion of the human-readable syntax to generate a generic representation of the material contained in the file; and

generating a manifest based on the generic representation, the generation comprising:

including in said manifest an identification of an executable module and an indication that either:

said executable module may be loaded into said address space; or

said executable module may not be loaded into said address space.

13. (Canceled)

14. (Previously Presented) The computer-readable storage medium of claim 12, wherein said rogue module is located external to the manifest and is operative to perform an unauthorized operation in said address space of said computer.

15-16. (Canceled)

17. (Previously Presented) A method of automating the generation of a manifest that governs the execution of a software object, the method comprising:

creating a file using a high-level description containing human-readable syntax that simplifies the describing of the manifest and permits a vendor to specify what may be loaded into an address space of a computer in which the software object is to be executed, the specification referring to one or more components that are external to the software and external to the specification;

parsing the file to generate an internal data structure representing a substance of the requirements with at least a portion of the human-readable syntax removed; and

using a manifest generation tool that accepts the internal data structure and automatically generates therefrom, the manifest, wherein the manifest generation tool does at least one of a) including, in said manifest, data from one of said one or more components; or b) computing a value based on one of said one or more components and including the computed value in said manifest, thereby eliminating the need for the vendor to manually insert the computed value into the manifest.

18. (Previously Presented) The method of claim 17, wherein said one or more components comprises a module, wherein said file indicates either that said module may be loaded into a secure address space or that said module may not be loaded into said address space, and wherein said manifest generation tool does at least one of:

including an identifier of said module in said manifest; or

computing a hash of said module and including the hash in said manifest.

19. (Previously Presented) The method of claim 17, wherein said one or more components comprise a key, wherein said file indicates either that modules signed with said key may be loaded into a secure address space or that modules signed with said key may not be loaded into said address space, and wherein said manifest generation tool retrieves said

key from a location identified in said file, and includes a certificate for said key in said manifest.

20. (Canceled)

21. (Previously Presented) The method of claim 17, wherein the method further comprises:

receiving a key associated with at least one of a) a vendor or b) a distributor of the software;

signing said manifest with said to produce a digital signature; and

including said digital signature in said manifest.

22. (Previously Presented) The method of claim 17, further comprising:

using a hardware security module to sign said manifest, said hardware security module being adapted to apply a key associated with at least one of a) a vendor or b) a distributor of the software without revealing said key outside said hardware security module.

23. (Currently Amended) A system comprising a processor for generating a manifest, the system <u>further</u> comprising:

a first parser implemented on the processor, the first parser configured to receive a manifest specification in a human-readable syntax indicative of requirements for a manifest, the first parser generating therefrom, a generic representation of said requirements by removing at least a portion of the human-readable syntax, said requirements relating to what may be loaded into an address space of a software object, said specification referring to one or more components external to said software and external to said specification; and

a first manifest generator that generates a manifest based on said representation and includes in said manifest information computed based on, said one or more

components, the manifest configured to interoperate with a security component that imposes a permeable barrier for selectively allowing acceptable modules to be loaded into the software space of the software object and blocking unacceptable modules from being loaded into the software space thereby preventing unauthorized tampering of the one or more components.

24. (Original) The system of claim 23, wherein said one or more components comprise a module, and wherein said first manifest generator generates said manifest by including, in said manifest, a datum that identifies said module.

25. (Previously Presented) The system of claim 24, wherein said datum comprises a hash of said module.

26. (Previously Presented) The system of claim 23, wherein said one or more components comprise a key, wherein said specification indicates either that acceptable modules signed with said key may be loaded into said address space or that unacceptable modules signed with said key may not be loaded into said address space, and wherein said first manifest generator retrieves said key from a file identified in said specification and includes said key in said manifest.

27. (Original) The system of claim 23, wherein said first manifest generator generates a digital signature for said manifest by signing said manifest with a key associated with a vendor or distributor of said software object, and includes said digital signature in said manifest.

28. (Canceled)

29. (Original) The system of claim 23, further comprising:

a second parser that receives a manifest specification indicative of requirements for a manifest, the second parser generating a representation of said requirements in the same format as said first parser,

wherein said first parser parses specifications in a first format and second parser parses specifications in a second format different from said first format, and wherein first manifest generator generates said manifest based on a representation produced either by said first parser or said second parser.

30. (Original) The system of claim 23, further comprising:

a second manifest generator that generates a manifest based on said representation, wherein said first manifest generator generates a manifest in a first format and second manifest generator generates a manifest in a second format different from said first format.

31. (Previously Presented) The method of claim 1, wherein at least one of the unacceptable modules is identified in the list by a version number.

32. (Previously Presented) The method of claim 1, wherein at least one of the unacceptable modules is identified in the list by a range of version numbers.

33. (Previously Presented) The computer-readable storage medium of claim 12, wherein the policy comprises an identity of an unacceptable module that is unconditionally barred from being executed in the address space of the software object.

34. (Previously Presented) The computer-readable storage medium of claim 33, wherein the unacceptable module is identified in the policy by a hash identifier.

35. (Previously Presented) The method of claim 1, wherein integrity of said address space is further enforced by confining inside said address space each of a) said software object, b) data used by said software object, and c) auxiliary code modules that are used by said software object.

36. (Previously Presented) The method of claim 1, wherein said MCF is an MCF file containing a high-level description using human-readable syntax and wherein parsing the MCF to create a generic representation comprises removing at least a portion of the human-readable syntax.

37. (Previously Presented) The method of claim 36, wherein removing at least a portion of the human-readable syntax comprises removing all human-readable syntax from the MCF.

38. (Previously Presented) The computer-readable storage medium of claim 12, wherein said file is written in a Manifest Configuration File (MCF) format, and wherein said manifest is generated in eXtensible Rights Markup Language (XRML) format.

39. (Previously Presented) The method of claim 17, wherein said file is written in a Manifest Configuration File (MCF) format, and wherein said manifest is generated in eXtensible Rights Markup Language (XRML) format.

40. (New) The method of claim 1, further comprising:

obtaining information pertaining to a set of public key certificates that are needed to verify correctness of the vendor-provided manifest back to a root of trust;

incorporating the obtained information into the MCF; and

using the MCF to include the set of public key certificates into the vendor-provided manifest, thereby eliminating the need to manually insert data corresponding to each individual public key certificate contained in the set of public key certificates.